Working with Nested Controls

It is possible to create controls as children of other controls in addition to so-called "top-level" controls, which are direct children of the dialog. Such controls are referred to as nested controls. The parent control is referred to as the container. We will also use the term siblings to refer to a set of child controls which all have the same parent. Clearly, there can be many different sets of sibling controls within a control hierarchy.

Creation of a control hierarchy enables the Natural programmer to group together controls such that they can be manipulated more easily and more efficiently within a Natural program. The following list describes the characteristics of nested controls:

- Their position is relative to the client area of the container control instead of relative to the dialog.
- Their display is clipped to their respective ancestor windows. This means that the areas of the nested control that are outside the boundary of its container are not visible. The Dialog Editor does not allow dragging of nested controls outside of the container.
- Nested controls are always displayed in front of their container control, regardless of their position in the control sequence.
- Nested controls are moved with their container control. This applies at both edit-time in the Dialog Editor (when the container is dragged) and at runtime (when the container’s RECTANGLE-X and/or RECTANGLE-Y attributes are modified).
- Nested controls are hidden when the container control is hidden, even though the VISIBLE attribute of the nested control remains unchanged.
- Nested controls are disabled at runtime when the container control is disabled, even though the ENABLED attribute of the nested control remains unchanged and even though the control does not become grayed.
- Nested controls are deleted when the container control is deleted.

Note:
Natural does not impose any arbitrary limits on the number of levels that a control hierarchy may contain. The level number for a particular control is displayed together with the control’s name in the Dialog Editor status bar combo box.

Which control types can be containers?

Not all control types are capable of acting as a container. It is not possible to create a control as a child of an input field, for example. There are currently three types of container control supported by Natural:

- Group frames that have the (new) ’container’ style set. This can be changed in the Dialog Editor (via its attributes window) after the group frame has been created. If a group frame is converted to a container, all controls that are spatially contained within it are moved in the control hierarchy to become descendants of the group frame. If a group frame is converted to a non-container, all direct children of the group frame are moved up a level in the hierarchy to become siblings of the group frame.
- ActiveX controls which are marked as "OLEMISC_SIMPLEFRAME" in the registry. This flag is fixed by design for a particular ActiveX control class.
- Control boxes. This control type is always a control container. Indeed, that is its entire purpose in life. See the section "Working with Control Boxes" for more information.

Creating a nested control

Nested controls are created in the Dialog Editor in the same way as non-nested controls are. If, during control insertion, the initial left mouse button click is determined to be over a container control, the new control is created automatically by Natural as a child of that container. Even before the mouse button is clicked in insert mode, the Dialog Editor’s status bar is continually updated with the container-relative mouse co-ordinates as the mouse cursor traverses the dialog.

In addition, nested controls can be indirectly created within the Dialog Editor when converting group frames to containers as described above.

At runtime, nested controls can be created dynamically, via the PROCESS GUI ACTION ADD statement for the nested control, by specifying the PARENT attribute as the handle of the required container control instead of the handle of the dialog. The nested control’s position (RECTANGLE-X and RECTANGLE-Y attributes) should be specified relative to the container’s client area. The client area of a control is the internal area of a control, excluding frame components such as 3-D borders, single-pixel frames resulting from used of the ’Framed’ style, and a control’s scroll bars.

Multiple selection, control sequence and clipboard operations

The Dialog Editor prohibits selection of multiple controls which do not have the same parent (i.e., are not all siblings of each other). This applies regardless of whether multiple controls are selected via "rubber banding" (marking of a region with the left mouse button held down) or via extended selection (holding down the <Shift> key whilst selecting a control). However, if a selected container control is deleted, then all its direct and indirect children (*descendants*) are of course implicitly deleted also, even though they are not explicitly selected. For this reason, a clipboard cut operation always copies the selected control(s) AND all descendant controls (if any) to the clipboard. For a clipboard copy operation, it is not clear whether to copy the container alone, or the container plus all its descendants. In this case, a message box is displayed, allowing the user to choose between the two options.

The pasting of controls from the clipboard uses the same control sequence (tab order) insertion position logic as for a control created from scratch. In both cases, the new control is created at a position in the control sequence immediately following the selected sibling (if any) plus any of its successive descendants. If a control other than a sibling is selected, an "effective sibling" is used instead, based on the position of the (active) selected control in the control sequence. The "active" selected control is the selected control (if any) which is highlighted using black (rather than gray) selection handles. If no selection is active, the control is inserted into the control sequence immediately preceding the first sibling control, or immediately after its container (or at the front of the control sequence for top-level controls) if the container is empty. Note, however, that the control sequence is maintained independently of the hierarchy. After a control has been created, it is possible to explicitly move any control to any position in the control sequence via the Control Sequence option on the Dialog menu.

The position of the newly-created control in the hierarchy is determined slightly differently in these two cases. In the case of a control being created from scratch, the container is determined by searching for the (topmost) container at the position where the left mouse button was pressed. However, in the case of pasting from the clipboard, we have no (X, Y)-position which we can use. In this case, the container is assumed to be the container of the selected control(s), or the dialog itself if no controls are selected. This means that if, for example, it is desired to copy and paste a control from one container to another, a control within the second container must be selected prior to the paste, not the container itself. If the second container is empty, this requires temporary creation of a dummy child control first, which can be deleted after the paste operation is complete.

Deletion of controls also deletes any of their descendant controls.

With the introduction of nested controls, the ’Select All’ command has been changed to operate in the following manner:

- If no control is currently selected, the command selects all top-level controls.
- Otherwise, all other controls that are siblings of the currently selected one(s) are additionally selected.

Thus, in the common case where only one level of hierarchy is in use, the ’Select All’ command continues, as before, to select all dialog controls.

Back to Event-Driven Programming Techniques.